Professionelle Softwareentwicklung

Woche 10

Projekt

IMPORTANT	Die Abgabe erfolgt über AUAS (https://auas.cs.uni-duesseldorf.de) Abgabefrist: 29.06.2018 um 13:00 Uhr (Ortszeit Düsseldorf)

Ihre Aufgabe ist die Implementierung der automatischen Abgabenverteilung des AUAS Systems. Sie erhalten eine startfähige Spring Boot Web-Anwendung, die schon das Rahmenwerk implementiert.

Ergänzung (20.6)

Das User Interface der Anwendung erlaubt es, die Zuordnung für die Übungsblätter in beliebiger Reihenfolge auszuführen. In der Realität ist die Zuordnung für vorangehende Blätter aber natürlich abgeschlossen, d.h. die Blätter werden in der richtigen Reihenfolge zugeordnet. Sie können in Ihren Tests davon ausgehen, dass alle Blätter, die als vorangegangene Blätter als Parameter in die Methode `abgabenZuordnen` übergeben werden bereits zugeordnet sind.

Starten der Anwendung

Die Anwendung kann aus der IDE heraus oder mit `gradle bootRun` gestartet werden. Es wird ein Webserver auf dem lokalen Port 8080 gestartet. Sollten Sie bereits eine Anwendung auf Ihrem Rechner ausführen, die Port 8080 belegt, können Sie den Port in der Datei `application.properties` umkonfigurieren. Nachdem der Server gestartet wurde, können Sie in Ihrem Browser die Anwendung unter `http://localhost:8080` oder `http://127.0.0.1:8080` aufrufen.

In der Anwendung finden Sie im unteren Teil eine Liste der Korrektorinnen [1: Ich verwende in der Aufgabenbeschreinung das generische Femininum.] und der wöchentlichen Stunden. Im oberen Teil finden Sie die Übungsblätter. Mit dem ersten Button können Sie ansehen, wieviele Übungsblätter auf welche Korrektorin verteilt wurden. Der zweite Button startet die automatische Verteilung der Abgaben auf Korrektorinnen, der dritte Button fügt neue Abgaben hinzu, der dritte Button funktioniert auch noch nachdem die automatische Verteilung ausgeführt wurde.

Aufgabe

Der vorgegebene Code ruft beim Drücken des Buttons "Automatisch zuordnen" die Methode `abgabenZuordnen` der Klasse `de.hhu.propra1.auas.domain.ZuordnungsService` auf. Der dort vorhandene Beispielcode verteilt alle Abgaben unfairerweise auf die erste Korrektorin. Sie sollen hier eine faire Verteilung implementieren, die folgende Kriterien erfüllt:

1. Die Anzahl der Abgaben, die eine Korrektorin zugeordnet bekommt ist proportional zu ihren Arbeitsstunden.
2. Es können nachträglich beliebig oft Abgaben hinzugefügt werden, diese können per Klick auf den Button "Automatisch zuordnen" zugeordnet werden. Eine einmal vorgenommene Zuordnung wird dabei niemals geändert.
3. Es wird Korrektorinnen geben, die überproportional viele Abgaben erhalten. Wir bezeichnen

das als Überhang. Es soll über die Zeit eine faire Zuordnung der Überhänge geben, d.h., es darf keine Korrektorin systematisch benachteiligt werden. Bei der Zuordnung müssen Sie also die vorangegangenen Zuordnungen mit berücksichtigen.

Ihr Code sollte die Regeln die Sie über guten Code kennengelernt haben befolgen (Regeln aus Projekt 2 und soweit anwendbar die SOLID Prinzipien und Low Coupling/High Cohesion). Der von Ihnen geschriebene Code muss eine Testabdeckung von 100% mit **sinnvollen** Tests erreichen. Sollten Sie aus gutem Grund die 100%ige Abdeckung nicht erreichen, müssen Sie das dokumentieren.

Testabdeckung

Es wird empfohlen, selber die Abdeckung Ihres Code durch Tests zu messen. Dazu können Sie in IntelliJ das eingebaute Feature "Run with Coverage" verwenden. Wenn Sie Eclipse benutzen, können Sie das Plugin EclEmma über den Marktplatz installieren, danach gibt es auch dort ein Starten mit Coverage-Analyse. Sie können die Analyse auch mit Hilfe des Jacoco Plugins direkt in Gradle verwenden. Das mitgelieferte Gradle Script ist bereits konfiguriert.

Formalia

Die Abgabe muss eine Zip-Datei sein. Es müssen folgende Bestandteile vorhanden sein:

1. Die Spring Boot Anwendung und die zugehörigen Tests.
2. Nach dem Entpacken der zip Datei muss die Anwendung mit `gradle bootRun` startbar sein, das ist bei der Vorlage bereits gegeben.
3. Eine Dokumentation, in der Sie beschreiben, wo Sie Schwierigkeiten hatten und wie Sie diese gelöst haben.

Ihre Abgabe darf auf **keinen Fall** generierte Dateien beinhalten, z.B. jar-Dateien oder .class-Files. Die dürfen auch nicht Quelltexte mitliefern, die aus den notwendige(n) Bibliothek(en) stammen.

Videos

Es ist recht einfach, trotz einer vollständigen Abdeckung alle Codezeilen Fehler im Code zu haben. Ich demonstriere das in einem Video. Sie sollten derartige Metriken, obwohl sie sicherlich das Vertrauen in den Code steigern, immer auch mit etwas Vorsicht betrachten und kritisch hinterfragen.

In einem weiteren Video demonstriere ich, wie Sie mit Hilfe von Dependency Inversion und Injection auch Code testen können, der Eingabe und Ausgabe beinhaltet.

Zusatzmaterial zur Veranstaltung

Video zum Thema JUnit Rules von Jens Schauder.